MATH 135 B PROJECT2

March 2024

1 CODE

```
#include <iostream>
#include <iomanip>

using namespace std;

// Define the function f(t, x)
double f(double t, double x) {
    return -4 * x + 1;
}

// Adams-Bashforth-Moulton predictor-corrector scheme
void adamsBashforthMoulton(double t0, double x0, double h, double T, int n) {
    double t = t0;
    double x = x0;
    double x_pred, x_corr;
    double t_prev[4] = {0}, x_prev[4] = {x0};

    cout << fixed << setprecision(20); // Set precision to 20 decimal places

    cout << "Adams-Bashforth-Moulton:" << endl;


    for (int i = 1; i <= n; i++) {
        t_prev[3] = t_prev[2];
        t_prev[2] = t_prev[1];
        t_prev[1] = t_prev[0];
        t_prev[0] = t;

        x_prev[3] = x_prev[2];
        x_prev[2] = x_prev[1];
        x_prev[1] = x_prev[0];
        x_prev[0] = x;
```

```
        t += h;

        x_pred = x + h / 24 * (55 * f(t_prev[0], x_prev[0]) - 59 * f(t_prev[1], x_prev[1]) +
                               37 * f(t_prev[2], x_prev[2]) - 9 * f(t_prev[3], x_prev[3]));

        x_corr = x + h / 24 * (9 * f(t, x_pred) + 19 * f(t_prev[0], x_prev[0]) -
                               5 * f(t_prev[1], x_prev[1]) + f(t_prev[2], x_prev[2]));

        x = x_corr;
        // floating point precision, some t cannot always equal to T
        if (t >= T - h / 10) {
            cout << "t = " << t << ", x = " << x << endl;
        }
    }
}

// Runge-Kutta method
void rungeKutta(double t0, double x0, double h, double T, int n) {
    double t = t0;
    double x = x0;
    double k1, k2, k3, k4;

    cout << fixed << setprecision(20); // Set precision to 20 decimal places

    cout << "Runge-Kutta:" << endl;

    for (int i = 1; i <= n; i++) {
        k1 = h * f(t, x);
        k2 = h * f(t + h / 2, x + k1 / 2);
        k3 = h * f(t + h / 2, x + k2 / 2);
        k4 = h * f(t + h, x + k3);

        x += (k1 + 2 * k2 + 2 * k3 + k4) / 6;
        t += h;
        // floating point precision, some t cannot always equal to T
        if (t >= T - h / 10) {
            cout << "t = " << t << ", x = " << x << endl;
        }
    }
}

int main() {
    double t0 = 0; // t = 0
    double x = 0.5; //x(0) = 0.5
    int T = 5;
```

```
    double h1 = 0.5;
    double h2 = 0.1;
    double h3 = 0.01;
    double h4 = 0.001;

    int n1 = T / h1;
    int n2 = T / h2;
    int n3 = T / h3;
    int n4 = T / h4;

    cout << fixed << setprecision(15); // Set precision to 15 decimal places

    cout << "Adams-Bashforth-Moulton with h = 0.5:" << endl;
    adamsBashforthMoulton(t0, x, h1, T, n1);
    cout << endl;

    cout << "Adams-Bashforth-Moulton with h = 0.1:" << endl;
    adamsBashforthMoulton(t0, x, h2, T, n2);
    cout << endl;

    cout << "Adams-Bashforth-Moulton with h = 0.01:" << endl;
    adamsBashforthMoulton(t0, x, h3, T, n3);
    cout << endl;

    cout << "Adams-Bashforth-Moulton with h = 0.001:" << endl;
    adamsBashforthMoulton(t0, x, h4, T, n4);
    cout << endl;

    cout << "Runge-Kutta with h = 0.5:" << endl;
    rungeKutta(t0, x, h1, T, n1);
    cout << endl;

    cout << "Runge-Kutta with h = 0.1:" << endl;
    rungeKutta(t0, x, h2, T, n2);
    cout << endl;

    cout << "Runge-Kutta with h = 0.01:" << endl;
    rungeKutta(t0, x, h3, T, n3);
    cout << endl;

    cout << "Runge-Kutta with h = 0.001:" << endl;
    rungeKutta(t0, x, h4, T, n4);
    cout << endl;

    return 0;
}
```

2 OUTPUT

```
rongzhongye@rongzhongs-Air math135b lab2 % g++ -o main main.cpp
rongzhongye@rongzhongs-Air math135b lab2 % ./main
Adams-Bashforth-Moulton with h = 0.5:
Adams-Bashforth-Moulton:
t = 5.00000000000000000000, x = 18.80422573805942221270

Adams-Bashforth-Moulton with h = 0.1:
Adams-Bashforth-Moulton:
t = 4.99999999999999822364, x = 0.25000000044764758567

Adams-Bashforth-Moulton with h = 0.01:
Adams-Bashforth-Moulton:
t = 4.99999999999993782751, x = 0.25000000051650073107

Adams-Bashforth-Moulton with h = 0.001:
Adams-Bashforth-Moulton:
t = 5.00000000000000444089, x = 0.25000000051545501201

Runge-Kutta with h = 0.5:
Runge-Kutta:
t = 5.00000000000000000000, x = 0.25000423377195213348

Runge-Kutta with h = 0.1:
Runge-Kutta:
t = 4.99999999999999822364, x = 0.25000000051837051318

Runge-Kutta with h = 0.01:
Runge-Kutta:
t = 4.99999999999993782751, x = 0.25000000051528864509

Runge-Kutta with h = 0.001:
Runge-Kutta:
t = 5.00000000000000444089, x = 0.25000000051528831202
```

3 Find x(t)

Solve the linear system:

$$\begin{cases} x'(t) = -4x(t) + 1 \\ x(0) = 0.5 \end{cases}$$

We assume that $x(t) = ke^{rt} + C$ and solve for k, r, and C.

Because $x(0) = 0.5$:

$$ke^{0\cdot r} + C = \frac{1}{2} \implies k + C = \frac{1}{2}$$

Differentiating $x(t)$, we get:

$$x'(t) = kre^{rt} = -4x(t) + 1$$

Substituting $x'(t)$ and rearranging:

$$x(t) = \frac{kr}{4}e^{rt} + \frac{1}{4} = ke^{rt} + c$$

we get:

$$\begin{cases} c = \frac{1}{4} \\ k + c = \frac{1}{2} \end{cases}$$

Solving, we find $C = \frac{1}{4}$ and $k = \frac{1}{4}$.

Because $x'(t) = -4x(t)+1 = 1$, $x(t) = \frac{1}{4}e^{rt}+\frac{1}{4}$, $x'(t) = \frac{1}{4}re^{rt}$, then combine them, we have:

$$\frac{1}{4}re^{rt} = -4(\frac{1}{4}e^{rt} + \frac{1}{4}) + 1 \implies \frac{1}{4}re^{rt} = -e^{rt} \implies r = -4$$

Thus, $r = -4$. Therefore, the solution is:

$$x(t) = \frac{1}{4}e^{-4t} + \frac{1}{4}$$

4 Exact solution

$x(5) = 0.25000000051528842304$

```
#include <iostream>
#include <iomanip>
#include <cmath>

using namespace std;

double f(int x){
    return 0.25 * exp(-4 * x) + 0.25;
}

int main(){
    cout << "EXACT ANSWER IS :" << endl;
```

```
    cout << fixed << setprecision(20) << f(5) << endl;

    return 0;
}
```

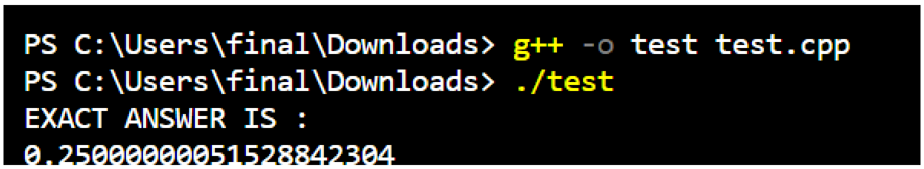

Figure 1: Enter Caption